Caesar’s Secret
Write-Up

By Isuka Kasthuriarachchi

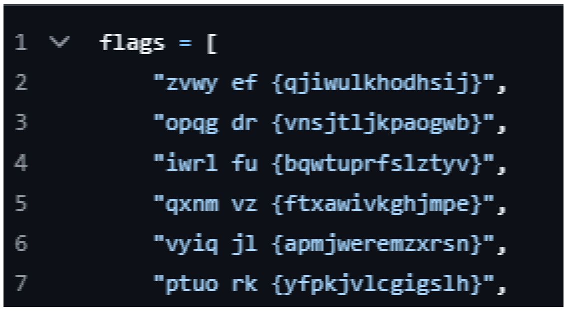
Challenge Description

We are given a file called resources.py which contains the following:

- flags: 50 encrypted flag candidates
- alphabet: a-z used for shifting
- dictionary: A list of common words

Step 1 - Inspecting the Data

Looking at resources.py, we can see entries like this:

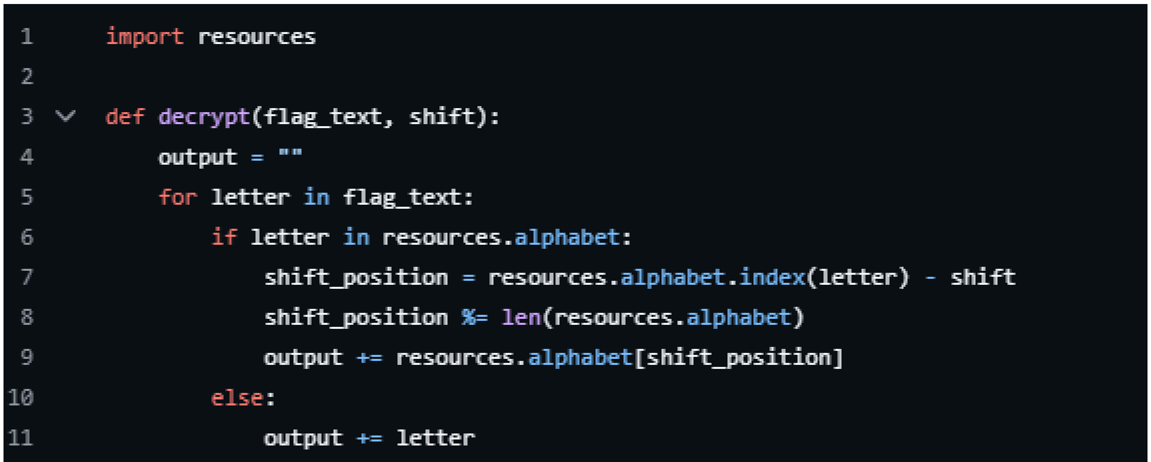


```
flags = [
    "zvwy ef {qjiwulkhodhsij}",
    "opqg dr {vnsjtljkpaogwb}",
    "iwrl fu {bqwtuprfslztyv}",
    "qxnm vz {ftxawivkghjmpe}",
    "vyiq jl {apmjweremzxrsn}",
    "ptuo rk {yfpkjvlcgigslh}",
```

Most of them are gibberish but one is the correct flag shifted by some unknown Caesar Cipher amount.

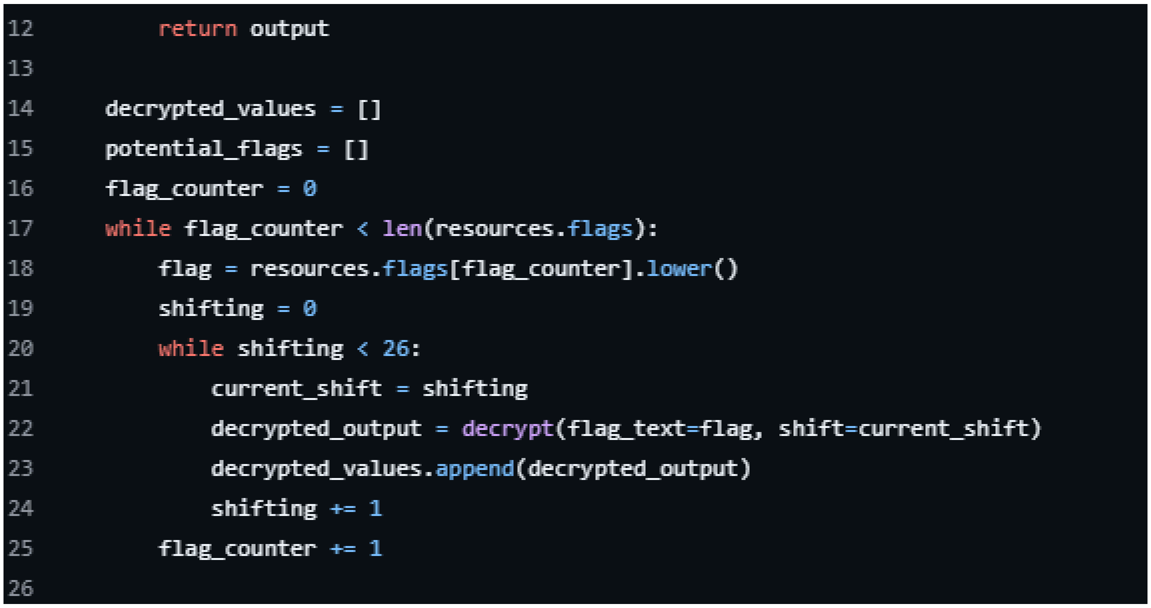
Step 2 - Decrypting the Flags

Now we have to write a script to loop through all the flags and decrypt them by trying all the possible shifts (0 to 25) and store all the possible decrypted values. The script looks something like this:



```
import resources

def decrypt(flag_text, shift):
    output = ""
    for letter in flag_text:
        if letter in resources.alphabet:
            shift_position = resources.alphabet.index(letter) - shift
            shift_position %= len(resources.alphabet)
            output += resources.alphabet[shift_position]
        else:
            output += letter
```



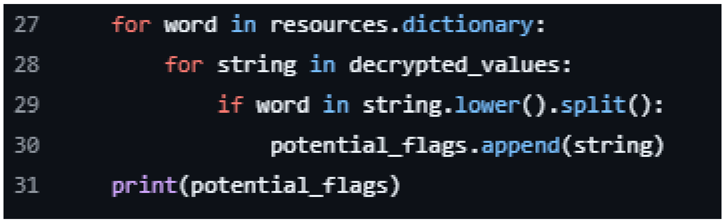
```
        return output

    decrypted_values = []
    potential_flags = []
    flag_counter = 0
    while flag_counter < len(resources.flags):
        flag = resources.flags[flag_counter].lower()
        shifting = 0
        while shifting < 26:
            current_shift = shifting
            decrypted_output = decrypt(flag_text=flag, shift=current_shift)
            decrypted_values.append(decrypted_output)
            shifting += 1
        flag_counter += 1

```

Step 3 - Writing the Brute Forcer and Filtering the Gibberish

We are also given a dictionary of some common words in the resources.py. We can use this to perform something similar to what is called a dictionary attack. We are going to check for those dictionary words within the decrypted outputs of the first script. This filters out most of the gibberish in the decrypted outputs. The brute forcer looks something like this:



```
    for word in resources.dictionary:
        for string in decrypted_values:
            if word in string.lower().split():
                potential_flags.append(string)
    print(potential_flags)
```

Step 4 - Manually Searching for the Flag

The ‘dictionary attack’ won’t give you the flag directly. It would only just narrow down your search. The next step is to manually look for the flag.

And that’s it!

You’ve got the flag!

Hope you had fun decrypting!